Patrones de Diseño: Factory y Observer

Irvin Alfonso solis FLores

Introducción

En este documento exploraremos dos patrones de diseño muy comunes en la Programación Orientada a Objetos: **Factory** y **Observer**.
Estos patrones ayudan a escribir código más **modular, reutilizable y desacoplado**.

Patrón Factory

El patrón **Factory** delega la creación de objetos a una clase especial llamada **Fábrica**.
La ventaja es que el código principal no necesita conocer los detalles de cómo se crean los objetos concretos.

Ejemplo real:
Imaginemos una aplicación de **servicio de transporte**, donde un usuario puede pedir un **Auto**, una **Moto** o una **Bicicleta** según su necesidad.
La fábrica se encarga de decidir qué objeto entregar.

Ejemplo en Python: Patrón Factory

1. Clases de transporte (productos concretos)

class Auto: def mover(self): return “Conduciendo un Auto ”

class Moto: def mover(self): return “Conduciendo una Moto ”

class Bicicleta: def mover(self): return “Pedaleando una Bicicleta ”

2. Clase Fábrica

class TransporteFactory: @staticmethod def crear_transporte(tipo): if tipo == “auto”: return Auto() elif tipo == “moto”: return Moto() elif tipo == “bicicleta”: return Bicicleta() else: raise ValueError(“Tipo de transporte no disponible”)

3. Uso en el código principal

pedido = “moto” transporte = TransporteFactory.crear_transporte(pedido) transporte.mover()

Ejemplo en Python: Patrón Observer

1. Clase Sujeto

class Clima: def **init**(self): self.observadores = [] self.estado = None

```
def agregar_observador(self, observador):
    self.observadores.append(observador)

def eliminar_observador(self, observador):
    self.observadores.remove(observador)

def notificar(self):
    for observador in self.observadores:
        observador.actualizar(self.estado)

def cambiar_estado(self, nuevo_estado):
    self.estado = nuevo_estado
    print(f"\n  El clima ha cambiado a: {self.estado}")
    self.notificar()
```

2. Observadores

class Usuario: def **init**(self, nombre): self.nombre = nombre

```
def actualizar(self, estado_clima):
    print(f"Notificación para {self.nombre}: El clima ahora es {estado_clima}")
```

3. Uso en el código principal

clima = Clima()

Usuarios se suscriben

usuario1 = Usuario(“Ana”) usuario2 = Usuario(“Luis”) clima.agregar_observador(usuario1) clima.agregar_observador(usuario2)

Cambio de estado del clima

clima.cambiar_estado(“Soleado ”) clima.cambiar_estado(“Lluvioso ”)